This is for LLVM 3.4.2 (with some differences for newer versions noted inline). The full manual is linked from the course website. There are often multiple variants of each of the following instructions; I list here only what I used (a sampling of what is available).

**Arithmetic**

```
<result> = add  <ty> <op1>, <op2>
<result> = mul  <ty> <op1>, <op2>
<result> = sdiv <ty> <op1>, <op2>
<result> = sub  <ty> <op1>, <op2>
```

**Boolean**

```
<result> = and <ty> <op1>, <op2>
<result> = or  <ty> <op1>, <op2>
<result> = xor <ty> <op1>, <op2>
```

**Comparison and Branching**

```
<result> = icmp <cond> <ty> <op1>, <op2>           ; e.g., <cond> = eq
br i1 <cond>, label <iftrue>, label <iffalse>
br label <dest>
```

**Loads & Stores**

```
<result> = load <ty>* <pointer>
    newer: <result> = load <ty>, <ty>* <pointer>
store <ty> <value>, <ty>* <pointer>
<result> = getelementptr <ty>* <ptrval>, i1 0, i32 <index>
    newer: <result> = getelementptr <ty>, <ty>* <ptrval>, i1 0, i32 <index>
```

**Invocation**

```
<result> = call <ty> <fnptrval>(<function args>)
    newer: <result> = call <ty> <fnval>(<function args>)
ret void
ret <ty> <value>
```

**Allocation**

```
<result> = alloca <ty>
```

**Miscellaneous**

```
<result> = bitcast <ty> <value> to <ty2>        ; cast type
<result> = trunc <ty> <value> to <ty2>          ; truncate to ty2
<result> = zext <ty> <value> to <ty2>           ; zero-extend to ty2
<result> = phi <ty> [<value 0>, <label 0>] [<value 1>, <label 1>] ... ; will revisit
```